An Integrated Architecture for Shallow and Deep Processing

Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Müller,
Günter Neumann, Jakub Piskorski, Ulrich Schäfer, Melanie Siegel, Hans Uszkoreit,
Feiyu Xu, Markus Becker and Hans-Ulrich Krieger
DFKI GmbH
Stuhlsatzenhausweg 3
Saarbrücken, Germany
whiteboard@dfki.de

Abstract

We present an architecture for the integration of shallow and deep NLP components which is aimed at flexible combination of different language technologies for a range of practical current and future applications. In particular, we describe the integration of a high-level HPSG parsing system with different high-performance shallow components, ranging from named entity recognition to chunk parsing and shallow clause recognition. The NLP components enrich a representation of natural language text with layers of new XML meta-information using a single shared data structure, called the text chart. We describe details of the integration methods, and show how information extraction and language checking applications for real-world German text benefit from a deep grammatical analysis.

1 Introduction

Over the last ten years or so, the trend in application-oriented natural language processing (e.g., in the area of term, information, and answer extraction) has been to argue that for many purposes, shallow natural language processing (SNLP) of texts can provide sufficient information for highly accurate and useful tasks to be carried out. Since the emergence of shallow techniques and the proof of their utility, the focus has been to exploit these technologies to the maximum, often ignoring certain complex issues, e.g. those which are typically well handled by deep NLP systems. Up to now, deep natural language processing (DNLP) has not played a significant role in the area of industrial NLP applications, since this technology often suffers from insufficient robustness and throughput, when confronted with large quantities of unrestricted text.

Current information extractions (IE) systems therefore do not attempt an exhaustive DNLP analysis of all aspects of a text, but rather try to analyse or “understand” only those text passages that contain relevant information, thereby warranting speed and robustness wrt. unrestricted NL text. What exactly counts as relevant is explicitly defined by means of highly detailed domain-specific lexical entries and/or rules, which perform the required mappings from NL utterances to corresponding domain knowledge. However, this “fine-tuning” wrt. a particular application appears to be the major obstacle when adapting a given shallow IE system to another domain or when dealing with the extraction of complex “scenario-based” relational structures. In fact, (Appelt and Israel, 1997) have shown that the current IE technology seems to have an upper performance level of less than 60% in such cases. It seems reasonable to assume that if a more accurate analysis of structural linguistic relationships could be provided (e.g., grammatical functions, referential relationships), this barrier might be overcome. Actually, the growing market needs in the wide area of intelligent information management systems seem to request such a break-through.

In this paper we will argue that the quality of current SNLP-based applications can be improved by integrating DNLP on demand in a focussed manner, and we will present a system that combines the fine-grained anaysis provided by HPSG parsing with a high-performance SNLP system into a generic and flexible NLP architecture.

1.1 Integration Scenarios

Owing to the fact that deep and shallow technologies are complementary in nature, integration is a non-trivial task: while SNLP shows its strength in the areas of efficiency and robustness, these aspects are problematic for DNLP systems. On the other hand, DNLP can deliver highly precise and fine-grained linguistic analyses. The challenge for integration is to combine these two paradigms according to their virtues.

Probably the most straightforward way to integrate the two is an architecture in which shallow and deep components run in parallel, using the results of DNLP, whenever available. While this kind of approach is certainly feasible for a real-time application such as Verbmobil, it is not ideal for processing large quantities of text: due to the difference in processing speed, shallow and deep NLP soon run out of sync. To compensate, one can imagine two possible remedies: either to optimize for precision, or for speed. The drawback of the former strategy is that the overall speed will equal the speed of the slowest component, whereas in case of the latter, DNLP will almost always time out, such that overall precision will hardly be distinguishable from a shallow-only system. What is thus called for is an integrated, flexible architecture where components can play at their strengths. Partial analyses from SNLP can be used to identify relevant candidates for the focussed use of DNLP, based on task or domain-specific criteria. Furthermore, such an integrated approach opens up the possibility to address the issue of robustness by using shallow analyses (e.g., term recognition) to increase the coverage of the deep parser, thereby avoiding a duplication of efforts. Likewise, integration at the phrasal level can be used to guide the deep parser towards the most likely syntactic analysis, leading, as it is hoped, to a considerable speed-up.

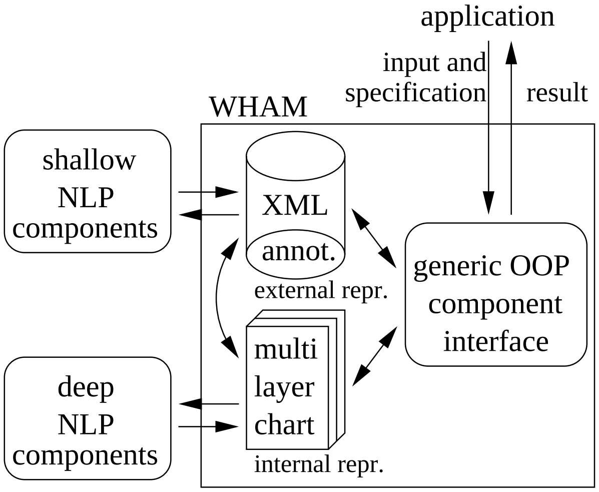

Figure 1: The WHITEBOARD architecture.

2 Architecture

The WHITEBOARD architecture defines a platform that integrates the different NLP components by enriching an input document through *XML annotations*. XML is used as a uniform way of representing and keeping all results of the various processing components and to support a transparent software infrastructure for LT-based applications. It is known that interesting linguistic information —especially when considering DNLP— cannot efficiently be represented within the basic XML markup framework ("typed parentheses structure"), e.g., linguistic phenomena like coreferences, ambiguous readings, and discontinuous constituents. The WHITEBOARD architecture employs a distributed multi-level representation of different annotations. Instead of translating all complex structures into one XML document, they are stored in different annotation layers (possibly non-XML, e.g. feature structures). Hyperlinks and "span" information together support efficient access between layers. Linguistic information of common interest (e.g. constituent structure extracted from HPSG feature structures) is available in XML format with hyperlinks to full feature structure representations externally stored in corresponding data files.

Fig. 1 gives an overview of the architecture of the WHITEBOARD Annotation Machine (WHAM). Applications feed the WHAM with input texts and a specification describing the components and configuration options requested. The core WHAM engine has an XML markup storage (external "offline" representation), and an internal "online" multi-level annotation chart (index-sequential access). Following the trichotomy of NLP data representation models in (Cunningham et al., 1997), the XML markup contains additive information, while the multi-level chart contains positional and abstraction-based information, e.g., feature structures representing NLP entities in a uniform, linguistically motivated form.

Applications and the integrated components access the WHAM results through an object-oriented programming (OOP) interface which is designed as general as possible in order to abstract from component-specific details (but preserving shallow and deep paradigms). The interfaces of the actually integrated components form subclasses of the generic interface. New components can be integrated by implementing this interface and specifying DTDs and/or transformation rules for the chart.

The OOP interface consists of iterators that walk through the different annotation levels (e.g., token spans, sentences), reference and seek operators that allow to switch to corresponding annotations on a different level (e.g., give all tokens of the current sentence, or move to next named entity starting from a given token position), and accessor methods that return the linguistic information contained in the chart. Similarily, general methods support navigating the type system and feature structures of the DNLP components. The resulting output of the WHAM can be accessed via the OOP interface or as XML markup.

The WHAM interface operations are not only used to implement NLP component-based applications, but also for the integration of deep and shallow processing components itself.

2.1 Components

2.1.1 Shallow NL component

Shallow analysis is performed by SPPC, a rule-based system which consists of a cascade of weighted finite–state components responsible for performing subsequent steps of the linguistic analysis, including: fine-grained tokenization, lexico-morphological analysis, part-of-speech filtering, named entity (NE) recognition, sentence boundary detection, chunk and subclause recognition, see (Piskorski and Neumann, 2000; Neumann and Piskorski, 2002) for details. SPPC is capable of processing vast amounts of textual data robustly and efficiently (ca. 30,000 words per second in standard PC environment). We will briefly describe the SPPC components which are currently integrated with the deep components.

Each token identified by a tokenizer as a potential word form is morphologically analyzed. For each token, its lexical information (list of valid readings including stem, part-of-speech and inflection information) is computed using a fullform lexicon of about 700,000 entries that has been compiled out from a stem lexicon of about 120,000 lemmas. After morphological processing, POS disambiguation rules are applied which compute a preferred reading for each token, while the deep components can back off to all readings. NE recognition is based on simple pattern matching techniques. Proper names (organizations, persons, locations), temporal expressions and quantities can be recognized with an average precision of almost 96% and recall of 85%. Furthermore, a NE–specific reference resolution is performed through the use of a dynamic lexicon which stores abbreviated variants of previously recognized named entities. Finally, the system splits the text into sentences by applying only few, but highly accurate contextual rules for filtering implausible punctuation signs. These rules benefit directly from NE recognition which already performs restricted punctuation disambiguation.

2.1.2 Deep NL component

The HPSG Grammar is based on a large–scale grammar for German (Müller, 1999), which was further developed in the VERBMOBIL project for translation of spoken language (Müller and Kasper, 2000). After VERBMOBIL the grammar was adapted to the requirements of the LKB/PET system (Copestake, 1999), and to written text, i.e., extended with constructions like free relative clauses that were irrelevant in the VERBMOBIL scenario.

The grammar consists of a rich hierarchy of 5,069 lexical and phrasal types. The core grammar contains 23 rule schemata, 7 special verb movement rules, and 17 domain specific rules. All rule schemata are unary or binary branching. The lexicon contains 38,549 stem entries, from which more than 70% were semi-automatically acquired from the annotated NEGRA corpus (Brants et al., 1999).

The grammar parses full sentences, but also other kinds of maximal projections. In cases where no full analysis of the input can be provided, analyses of fragments are handed over to subsequent modules. Such fragments consist of maximal projections or single words.

The HPSG analysis system currently integrated in the WHITEBOARD system is PET (Callmeier, 2000). Initially, PET was built to experiment with different techniques and strategies to process unification-based grammars. The resulting system provides efficient implementations of the best known techniques for unification and parsing.

As an experimental system, the original design lacked open interfaces for flexible integration with external components. For instance, in the beginning of the WHITEBOARD project the system only accepted fullform lexica and string input. In collaboration with Ulrich Callmeier the system was extended. Instead of single word input, input items can now be complex, overlapping and ambiguous, i.e. essentially word graphs. We added dynamic creation of atomic type symbols, e.g., to be able to add arbitrary symbols to feature structures. With these enhancements, it is possible to build flexible interfaces to external components like morphology, tokenization, named entity recognition, etc.

3 Integration

Morphology and POS The coupling between the morphology delivered by SPPC and the input needed for the German HPSG was easily established. The morphological classes of German are mapped onto HPSG types which expand to small feature structures representing the morphological information in a compact way. A mapping to the output of SPPC was automatically created by identifying the corresponding output classes.

Currently, POS tagging is used in two ways. First, lexicon entries that are marked as preferred by the shallow component are assigned higher priority than the rest. Thus, the probability of finding the correct reading early should increase without excluding any reading. Second, if for an input item no entry is found in the HPSG lexicon, we automatically create a default entry, based on the part–of–speech of the preferred reading. This increases robustness, while avoiding increase in ambiguity.

Named Entity Recognition Writing HPSG grammars for the whole range of NE expressions etc. is a tedious and not very promising task. They typically vary across text sorts and domains, and would require modularized subgrammars that can be easily exchanged without interfering with the general core. This can only be realized by using a type interface where a class of named entities is encoded by a general HPSG type which expands to a feature structure used in parsing. We exploit such a type interface for coupling shallow and deep processing. The classes of named entities delivered by shallow processing are mapped to HPSG types. However, some fine-tuning is required whenever deep and shallow processing differ in the amount of input material they assign to a named entity.

An alternative strategy is used for complex syntactic phrases containing NEs, e.g., PPs describing time spans etc. It is based on ideas from Explanation–based Learning (EBL, see (Tadepalli and Natarajan, 1996)) for natural language analysis, where analysis trees are retrieved on the basis of the surface string. In our case, the part-of-speech sequence of NEs recognised by shallow analysis is used to retrieve pre-built feature structures. These structures are produced by extracting NEs from a corpus and processing them directly by the deep component. If a correct analysis is delivered, the lexical parts of the analysis, which are specific for the input item, are deleted. We obtain a sceletal analysis which is underspecified with respect to the concrete input items. The part-of-speech sequence of the original input forms the access key for this structure. In the application phase, the underspecified feature structure is retrieved and the empty slots for the input items are filled on the basis of the concrete input.

The advantage of this approach lies in the more elaborate semantics of the resulting feature structures for DNLP, while avoiding the necessity of adding each and every single name to the HPSG lexicon. Instead, good coverage and high precision can be achieved using prototypical entries.

Lexical Semantics When first applying the original VERBMOBIL HPSG grammar to business news articles, the result was that 78.49% of the missing lexical items were nouns (ignoring NEs). In the integrated system, unknown nouns and NEs can be recognized by SPPC, which determines morphosyntactic information. It is essential for the deep system to associate nouns with their semantic sorts both for semantics construction, and for providing semantically based selectional restrictions to help constraining the search space during deep parsing. GermaNet (Hamp and Feldweg, 1997) is a large lexical database, where words are associated with POS information and semantic sorts, which are organized in a fine-grained hierarchy. The HPSG lexicon, on the other hand, is comparatively small and has a more coarse-grained semantic classification.

To provide the missing sort information when recovering unknown noun entries via SPPC, a mapping from the GermaNet semantic classification to the HPSG semantic classification (Siegel et al., 2001) is applied which has been automatically acquired. The training material for this learning process are those words that are both annotated with semantic sorts in the HPSG lexicon and with synsets of GermaNet. The learning algorithm computes a mapping relevance measure for associating semantic concepts in GermaNet with semantic sorts in the HPSG lexicon. For evaluation, we examined a corpus of 4664 nouns extracted from business news that were not contained in the HPSG lexicon. 2312 of these were known in GermaNet, where they are assigned 2811 senses. With the learned mapping, the GermaNet senses were automatically mapped to HPSG semantic sorts. The evaluation of the mapping accuracy yields promising results: In 76.52% of the cases the computed sort with the highest relevance probability was correct. In the remaining 20.70% of the cases, the correct sort was among the first three sorts.

3.1 Integration on Phrasal Level

In the previous paragraphs we described strategies for integration of shallow and deep processing where the focus is on improving DNLP in the domain of lexical and sub-phrasal coverage.

We can conceive of more advanced strategies for the integration of shallow and deep analysis at the level of phrasal syntax by guiding the deep syntactic parser towards a partial pre-partitioning of complex sentences provided by shallow analysis systems. This strategy can reduce the search space, and enhance parsing efficiency of DNLP.

length	coverage	complete match	LP	LR	0CB	≤2CB
≤40	100	80.4	93.4	92.9	92.1	98.9
>40	99.8	78.6	92.4	92.2	90.7	98.5

Training: 16,000 NEGRA sentences
Testing: 1,058 NEGRA sentences

Figure 2: Stochastic topological parsing: results

Stochastic Topological Parsing The traditional syntactic model of *topological fields* divides basic clauses into distinct fields: so-called *pre-*, *middle- and post-fields*, delimited by verbal or sentential markers. This topological model of German clause structure is underspecified or partial as to non-sentential constituent boundaries, but provides a linguistically well-motivated, and theory-neutral *macrostructure* for complex sentences. Due to its linguistic underpinning the topological model provides a pre-partitioning of complex sentences that is (i) highly compatible with deep syntactic structures and (ii) maximally effective to increase parsing efficiency. At the same time (iii) partiality regarding the constituency of non-sentential material ensures the important aspects of robustness, coverage, and processing efficiency.

In (Becker and Frank, 2002) we present a *corpus-driven stochastic topological parser* for German, based on a topological restructuring of the NEGRA corpus (Brants et al., 1999). For topological treebank conversion we build on methods and results in (Frank, 2001). The stochastic topological parser follows the probabilistic model of non-lexicalised PCFGs (Charniak, 1996). Due to abstraction from constituency decisions at the sub-sentential level, and the essentially POS-driven nature of topological structure, this rather simple probabilistic model yields surprisingly high figures of accuracy and coverage (see Fig.2 and (Becker and Frank, 2002) for more detail), while context-free parsing guarantees efficient processing.

The next step is to elaborate a (partial) mapping of shallow topological and deep syntactic structures that is maximally effective for preference-gui-

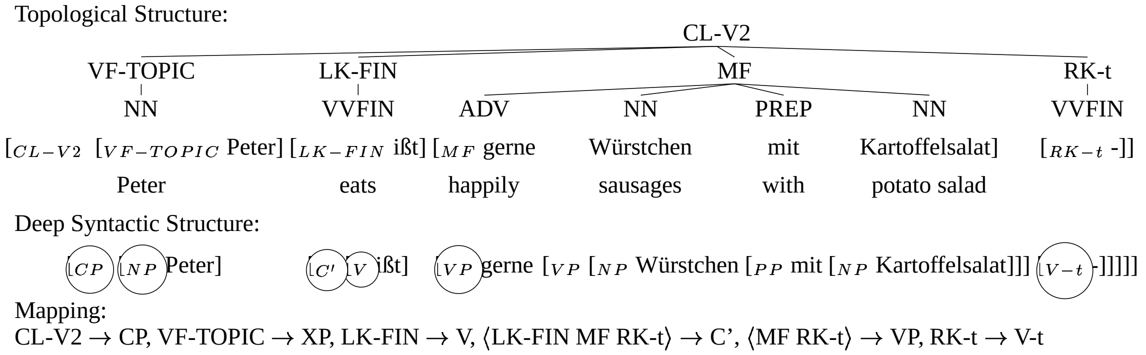
Topological Structure:
CL-V2
VF-TOPIC
LK-FIN
MF
RK-t
NN
VVFIN
ADV
NN
PREP
NN
VVFIN
$[_{CL-V2}$ $[_{VF-TOPIC}$ Peter] $[_{LK-FIN}$ ißt] $[_{MF}$ gerne Würstchen mit Kartoffelsalat] $[_{RK-t}$ -]]
Peter eats happily sausages with potato salad
Deep Syntactic Structure:
$[_{CP}$ $[_{NP}$ Peter] $[_{C'}$ $[_{V}$ ißt] $[_{VP}$ gerne $[_{VP}$ $[_{NP}$ Würstchen $[_{PP}$ mit $[_{NP}$ Kartoffelsalat]]] $[_{V-t}$ -]]]]]
Mapping:
CL-V2 → CP, VF-TOPIC → XP, LK-FIN → V, ⟨LK-FIN MF RK-t⟩ → C', ⟨MF RK-t⟩ → VP, RK-t → V-t

Figure 3: Matching topological and deep syntactic structures

ded deep syntactic analysis, and thus, efficiency improvements in deep syntactic processing. Such a mapping is illustrated for a verb-second clause in Fig.3, where matching constituents of topological and deep-syntactic phrase structure are indicated by circled nodes. With this mapping defined for all sentence types, we can proceed to the technical aspects of integration into the WHITEBOARD architecture and XML text chart, as well as preference-driven HPSG analysis in the PET system.

4 Experiments

An evaluation has been started using the NEGRA corpus, which contains about 20,000 newspaper sentences. The main objectives are to evaluate the syntactic coverage of the German HPSG on newspaper text and the benefits of integrating deep and shallow analysis. The sentences of the corpus were used in their original form without stripping, e.g. parenthesized insertions.

We extended the HPSG lexicon semi-automatically from about 10,000 to 35,000 stems, which roughly corresponds to 350,000 full forms. Then, we checked the lexical coverage of the deep system on the whole corpus, which resulted in 28.6% of the sentences being fully lexically analyzed. The corresponding experiment with the integrated system yielded an improved lexical coverage of 71.4%, due to the techniques described in section 3. This increase is not achieved by manual extension, but only through synergy between the deep and shallow components.

To test the syntactic coverage, we processed the subset of the corpus that was fully covered lexically (5878 sentences) with deep analysis only. The results are shown in table 4 in the second column. In order to evaluate the integrated system we processed 20,568 sentences from the corpus without further extension of the HPSG lexicon (see table 4, third column).

	Deep	Integrated
# sentences	20,568	
avg. sentence length	16.83	
avg. lexical ambiguity	2.38	1.98
avg. # analyses	16.19	18.53
analysed sentences	2,569	4,546
lexical coverage	28.6%	71.4%
overall coverage	12.5%	22.1%

Figure 4: Evaluation of German HPSG

About 10% of the sentences that were successfully parsed by deep analysis only could not be parsed by the integrated system, and the number of analyses per sentence dropped from 16.2% to 8.6%, which indicates a problem in the morphology interface of the integrated system. We expect better overall results once this problem is removed.

5 Applications

Since typed feature structures (TFS) in Whiteboard serve as both a representation and an interchange format, we developed a Java package (JTFS) that implements the data structures, together with the necessary operations. These include a lazy-copying unifier, a subsumption and equivalence test, deep copying, iterators, etc. JTFS supports a dynamic construction of typed feature structures, which is important for information extraction.

5.1 Information Extraction

Information extraction in Whiteboard benefits both from the integration of the shallow and deep analysis results and from their processing methods. We chose *management succession* as our application domain. Two sets of template filling rules are defined: pattern-based and unification-based rules. The pattern-based rules work directly on the output delivered by the shallow analysis, for example,

(1) Nachfolger von $\boxed{1}\mathit{person\_name} \longrightarrow$ $[\texttt{person\_out}\ \boxed{1}]$.

This rule matches expressions like *Nachfolger von Helmut Kohl* (successor of) which contains two string tokens *Nachfolger* and *von* followed by a person name, and fills the slot of person_out with the recognized person name *Helmut Kohl*. The pattern-based grammar yields good results by recognition of local relationships as in (1). The unification-based rules are applied to the deep analysis results. Given the fine-grained syntactic and semantic analysis of the HPSG grammar and its robustness (through SNLP integration), we decided to use the semantic representation (MRS, see (Copestake et al., 2001)) as additional input for IE. The reason is that MRSs express precise relationships between the chunks, in particular, in constructions involving (combinations of) free word order, long distance dependencies, control and raising, or passive, which are very difficult, if not impossible, to recognize for a pattern-based grammar. E.g., the short sentence (2) illustrates a combination of free word order, control, and passive. The subject of the passive verb *wurde gebeten* is located in the middle field and is at the same time the subject of the infinitive verb *zu übernehmen*. A deep (HPSG) analysis can recognize the dependencies quite easily, whereas a pattern based grammar cannot determine, e.g., for which verb *Peter Miscke* or *Dietmar Hopp* is the subject.

(2) Peter Miscke following was Dietmar Hopp asked, the development_sector to take_over.

Peter *Miscke*
Entwicklungsabteilung zu
zufolge *wurde* *Dietmar* *Hopp*
übernehmen.
gebeten, *die*

“ According to Peter Miscke, Dietmar Hopp was asked to take over the development sector.”

We employ typed feature structures (TFS) as our modelling language for the definition of scenario template types and template element types. Therefore, the template filling results from shallow and deep analysis can be uniformly encoded in TFS. As a side effect, we can easily adapt JTFS unification for the template merging task, by interperting the partially filled templates from deep and shallow analysis as constraints. E.g., to extract the relevant information from the above sentence, the following unification-based rule can be applied:

$$\begin{bmatrix} \text{PERSON\_IN } \boxed{2} \\ \text{DIVISION } \boxed{3} \\ \text{MRS } \begin{bmatrix} \text{PRED “übernehmen”} \\ \text{AGENT } \boxed{2} \\ \text{THEME } \boxed{3} \end{bmatrix} \end{bmatrix}$$

5.2 Language checking

Another area where DNLP can support existing shallow-only tools is grammar and controlled language checking. Due to the scarce distribution of true errors (Becker et al., to appear), there is a high a priori probability for false alarms. As the number of false alarms decides on user-acceptance, precision is of utmost importance and cannot easily be traded for recall. Current controlled language checking systems for German, such as MULTILINT (http://www.iai.uni-sb.de/en/multien.html) or FLAG (http://flag.dfki.de), build exclusively on SNLP: while checking of local errors (e.g. NP-internal agreement, prepositional case) can be performed quite reliably by such a system, error types involving non-local dependencies, or access to grammatical functions are much harder to detect. The use of DNLP in this area is confronted with several systematic problems: first, formal grammars are not always available, e.g., in the case of controlled languages; second, erroneous sentences lie outside the language defined by the competence grammar, and third, due to the sparse distribution of errors, a DNLP system will spend most of the time parsing perfectly well-formed sentences. Using an integrated approach, a shallow checker can be used to cheaply identify initial error candidates, while false alarms can be eliminated based on the richer annotations provided by the deep parser.

6 Discussion

In this paper we reported on an implemented system called WHITEBOARD which integrates different shallow components with a HPSG–based deep system. The integration is realized through the metaphor of textual annotation. To best of our knowledge, this is the first implemented system which integrates high-performance shallow processing with an advanced deep HPSG–based analysis system. There exists only very little other work that considers integration of shallow and deep NLP using an XML–based architecture, most notably (Grover and Lascarides, 2001). However, their integration efforts are largely limited to the level of POS tag information.

Acknowledgements

This work was supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project WHITEBOARD, FKZ: 01 IW 002. Special thanks to Ulrich Callmeier for his technical support concerning the integration of PET.

References

D. Appelt and D. Israel. 1997. *Building information extraction systems*. Tutorial during the 5th ANLP, Washington.

M. Becker and A. Frank. 2002. A Stochastic Topological Parser of German. In *Proceedings of COLING 2002*, Teipei, Taiwan.

M. Becker, A. Bredenkamp, B. Crysmann, and J. Klein. to appear. Annotation of error types for german newsgroup corpus. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer, Dordrecht.

T. Brants, W. Skut, and H. Uszkoreit. 1999. Syntactic Annotation of a German newspaper corpus. In *Proceedings of the ATALA Treebank Workshop*, pages 69–76, Paris, France.

U. Callmeier. 2000. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):99 – 108.

E. Charniak. 1996. Tree-bank Grammars. In *AAAI-96. Proceedings of the 13th AAAI*, pages 1031–1036. MIT Press.

A. Copestake, A. Lascarides, and D. Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France.

A. Copestake. 1999. The (new) LKB system. ftp://www-csli.stanford.edu/˜aac/newdoc.pdf.

H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. 1997. Software Infrastructure for Natural Language Processing. In *Proceedings of the Fifth ANLP*, March.

A. Frank. 2001. Treebank Conversion. Converting the NEGRA Corpus to an LTAG Grammar. In *Proceedings of the EUROLAN Workshop on Multi-layer Corpus-based Analysis*, pages 29–43, Iasi, Romania.

C. Grover and A. Lascarides. 2001. XML-based data preparation for robust deep parsing. In *Proceedings of the 39th ACL*, pages 252–259, Toulouse, France.

B. Hamp and H. Feldweg. 1997. Germanet - a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid.

S. Müller and W. Kasper. 2000. HPSG analysis of German. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, Artificial Intelligence, pages 238–253. Springer-Verlag, Berlin Heidelberg New York.

S. Müller. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Max Niemeyer Verlag, Tübingen.

G. Neumann and J. Piskorski. 2002. A shallow text processing core engine. *Computational Intelligence*, to appear.

J. Piskorski and G. Neumann. 2000. An intelligent text extraction and navigation system. In *Proceedings of the RIAO-2000*. Paris, April.

M. Siegel, F. Xu, and G. Neumann. 2001. Customizing germanet for the use in deep linguistic processing. In *Proceedings of the NAACL 2001 Workshop* WordNet and Other Lexical Resources: Applications, Extensions and Customizations, Pittsburgh,USA, July.

P. Tadepalli and B. Natarajan. 1996. A formal framework for speedup learning from problems and solutions. *Journal of AI Research*, 4:445 – 475.